Learning the Linux Kernel Configuration Space:

Results and Challenges

Prof. Mathieu Acher (University of Rennes 1, Inria)

@acherm

THE LINUX FOUNDATION

Preprints (feedbacks welcome!)

DiverSE
Diversity-Centric Software Engineering

- Learning From Thousands of Build Failures of Linux Kernel Configurations
  - Mathieu Acher, Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Djamel Eddine Khelladi, Jean-Marc Jézéquel
  - https://hal.inria.fr/hal-02147012
- Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes
  - Mathieu Acher, Hugo Martin, Juliana Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
  - https://hal.inria.fr/hal-02314830

Linux everywhere since

highly configurable

```
config X86_X2APIC
        bool "Support x2apic"
        depends on X86_LOCAL_APIC && X86_64 && (IRQ_REMAP || HYPERVISOR_GUEST)
        ---help---
          This enables x2apic support on CPUs that have this feature.

          This allows 32-bit apic IDs (so it can support very large systems),
```

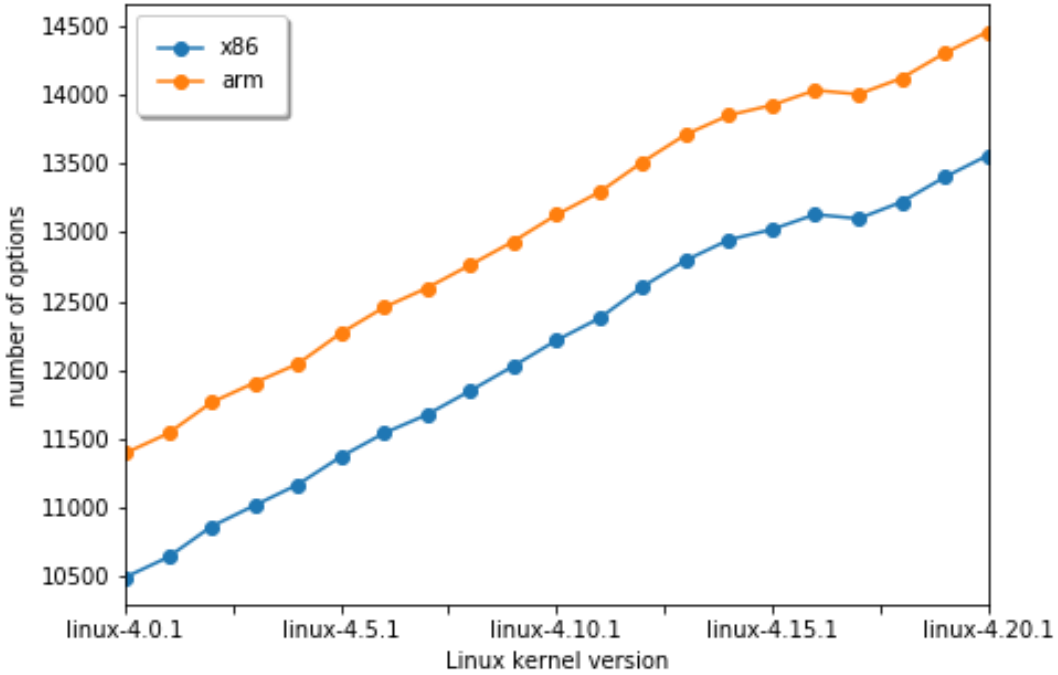
```
config IOSF_MBI
        tristate "Intel SoC IOSF Sideband support for SoC platforms"
        depends on PCI
        ---help---
          This option enables sideband register access support for Intel SoC
          platforms. On these platforms the IOSF sideband is used in lieu of
          MSR's for some register accesses, mostly but not limited to thermal
          and power. Drivers may query the availability of this device to
          determine if they need the sideband in order to work on these
          platforms. The sideband is available on the following SoC products.
```

Kconfig files/doc

```
#
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

.config

15,000+ options



How to ensure that all Linux kernel configurations build/boot?

Many failures are due to buggy (combinations of) options

Developers/Users can hardly **test** all possible variants:

- Static analysis
- Actual builds and kernel instrumentation

(e.g., 0-day/kbuild robot/KernelCI)

Given a kernel configuration, what's its size/boot time?

```
#
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

Who knows what's the effect of options?

Default configurations/options' values

Documentation

Smart configurators

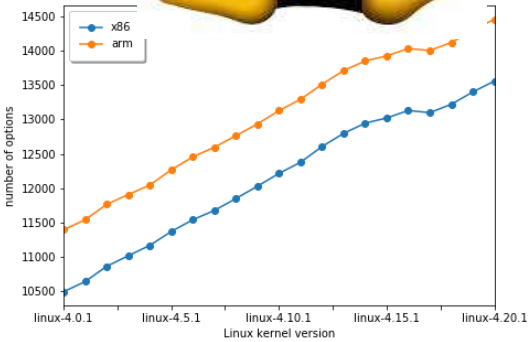
Bugs (unintended effects)

Effects of (combinations of) options on build status/boot/size/boot time/ performance/securi

```
#
# Processor type and features
#
# CONFIG_ZONE_DMA is not set
# CONFIG_SMP is not set
# CONFIG_X86_FEATURE_NAMES is not set
# CONFIG_X86_FAST_FEATURE_TESTS is not set
CONFIG_X86_X2APIC=y
CONFIG_X86_MPPARSE=y
CONFIG_GOLDFISH=y
# CONFIG_INTEL_RDT_A is not set
# CONFIG_X86_EXTENDED_PLATFORM is not set
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

General problem:

Learning the configuration space out of a sample of observations



15,000+ options

TRISTATE	61.72
BOOL	36.17
INT	1.63
STRING	0.33
HEX	0.15

Linux 4.15 (% of types' options)

3^{9000}

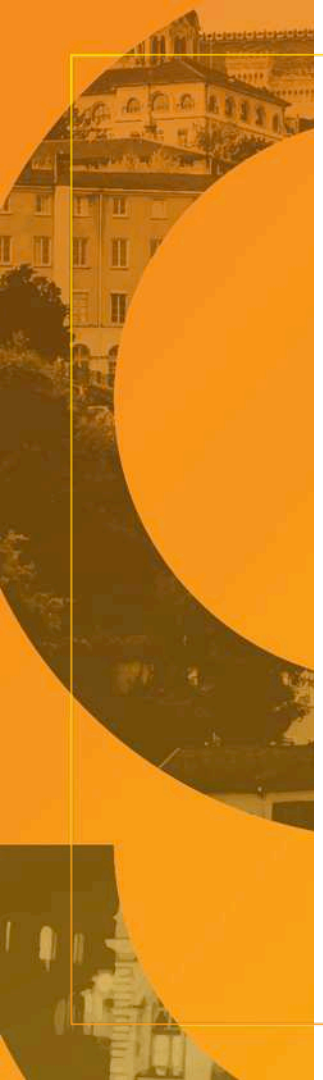
2^{6000}

$\approx 10^{6000}$ configurations

Linux Kernel

$\approx 10^{6000}$

configurations



Linux Kernel

$\approx 10^{6000}$ configurations

$\approx 10^{80}$ is the estimated number of atoms in the universe

$\approx 10^{40}$ is the estimated number of possible chess positions

Linux vs AlphaZero

Building a kernel configuration takes 10 minutes on average on a recent machine

Trial and error is cheap for chess/Go, you can experience winning/losing billions of time

Linux vs AlphaZero

In chess/Go, you can fully observe the outcome, without noise and with a perfect simulator

Think about technically measuring the boot time of a kernel out of a configuration

Is taming the Linux kernel configuration space a harder problem than resolving Chess?

# configurations	$\approx 10^{6000}$	$\approx 10^{40}$
exploration	costly and hard to engineer	cheap with a perfect simulator

You cannot build ≈10^{6000} configurations.

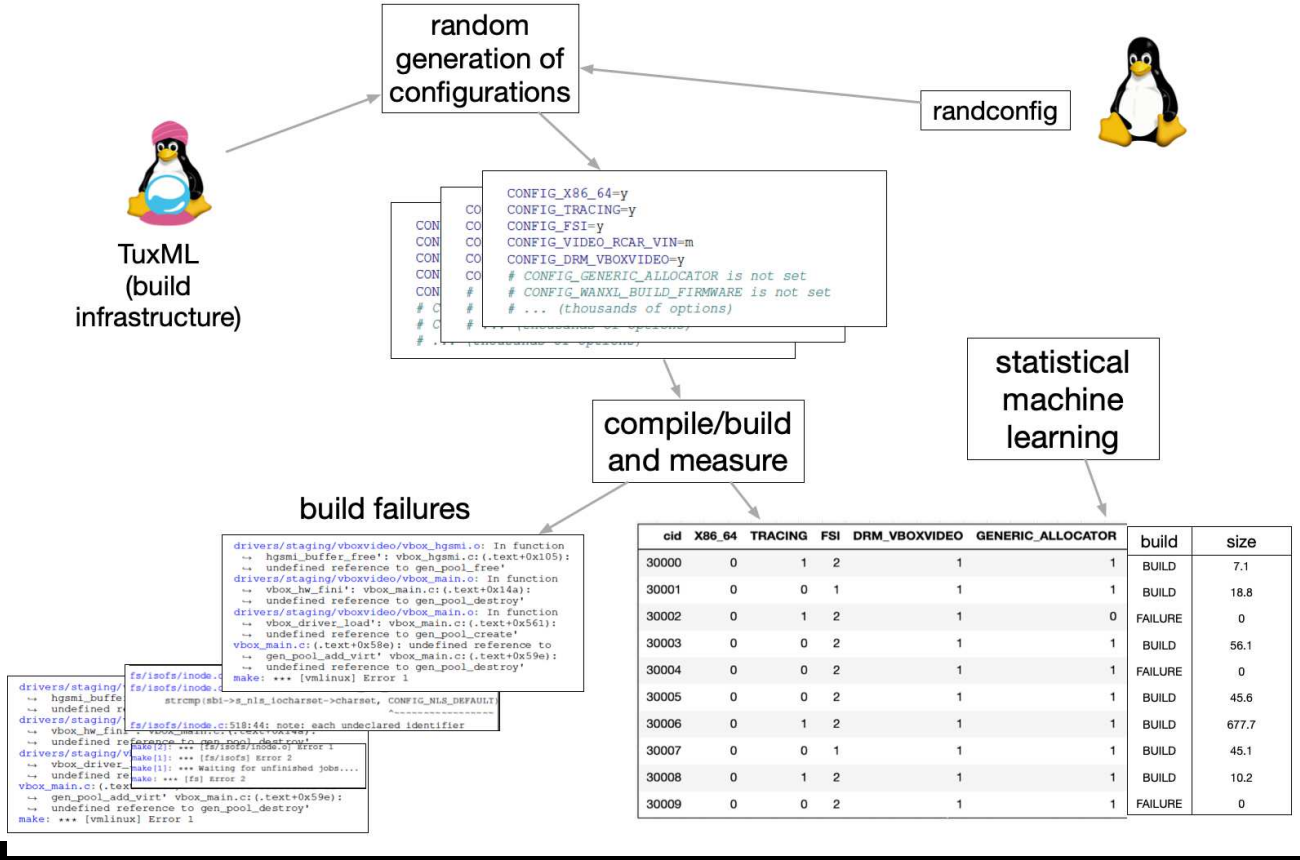
TUXML: predicting out of a (small) sample kernels' properties (e.g., size)

```
#
# Processor type and featu
#
# CONFIG_ZON
# CONFIG_SMP
# CONFIG_X86
# CONFIG_X86
CONFIG_X86_X
CONFIG_X86_M
CONFIG_GOLDF
# CONFIG_INT
# CONFIG_X86
CONFIG_IOSF_MBI=m
CONFIG_IOSF_MBI_DEBUG=y
CONFIG_X86_SUPPORTS_MEMORY_FAILURE=y
# CONFIG_SCHED_OMIT_FRAME_POINTER is not set
```

Plan

- Sampling and Learning with TUXML
- Results over 150K+ configurations
  - build failure understanding and prevention
  - kernel size prediction
- Challenges
  - “smart” build infrastructure
  - with devs/contributors in the loop

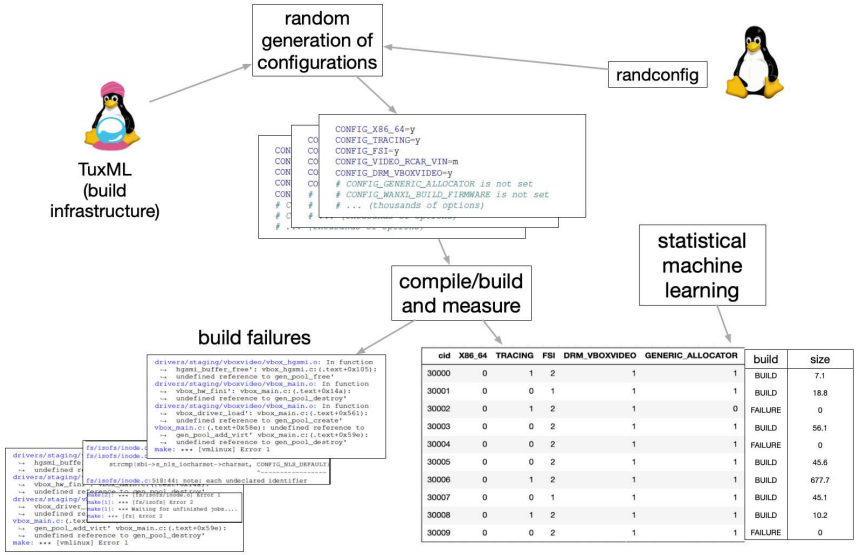
TUXML: Sampling, Measuring, Learning



cid	X86_64	TRACING	FSI	DRM_VBOXVIDEO	GENERIC_ALLOCATOR	build	size
30000	0	1	2	1	1	BUILD	7.1
30001	0	0	1	1	1	BUILD	18.8
30002	0	1	2	1	0	FAILURE	0
30003	0	0	2	1	1	BUILD	56.1
30004	0	0	2	1	1	FAILURE	0
30005	0	0	2	1	1	BUILD	45.6
30006	0	1	2	1	1	BUILD	677.7
30007	0	0	1	1	1	BUILD	45.1
30008	0	1	2	1	1	BUILD	10.2
30009	0	0	2	1	1	FAILURE	0

TUXML: Sampling, Measuring, Learning

https://github.com/TuxML/ProjetIrma/



Docker for a reproducible environment with tools/packages needed and Python procedures inside

Easy to launch campaign:
”python3 kernel_generator.py 10”

builds/measures
10 random configurations
(information sent to a database)

TUXML: Sampling, Measuring, Learning

https://github.com/TuxML/ProjetIrma/

cid	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24,3 Kio]	[BLOB - 33,7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzImage : 8855504 , GZIP-vmlinux : 10943304 ,...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25,8 Kio]	[BLOB - 34,7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24,2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 16452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24,1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
[illegible]	[illegible]:03	305.705	[BLOB - 26,1 Kio]	[BLOB - 28,8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16852088

Console de requêtes SQL

(information sent to a database)

Data: Linux version 4.13.3 and 4.15

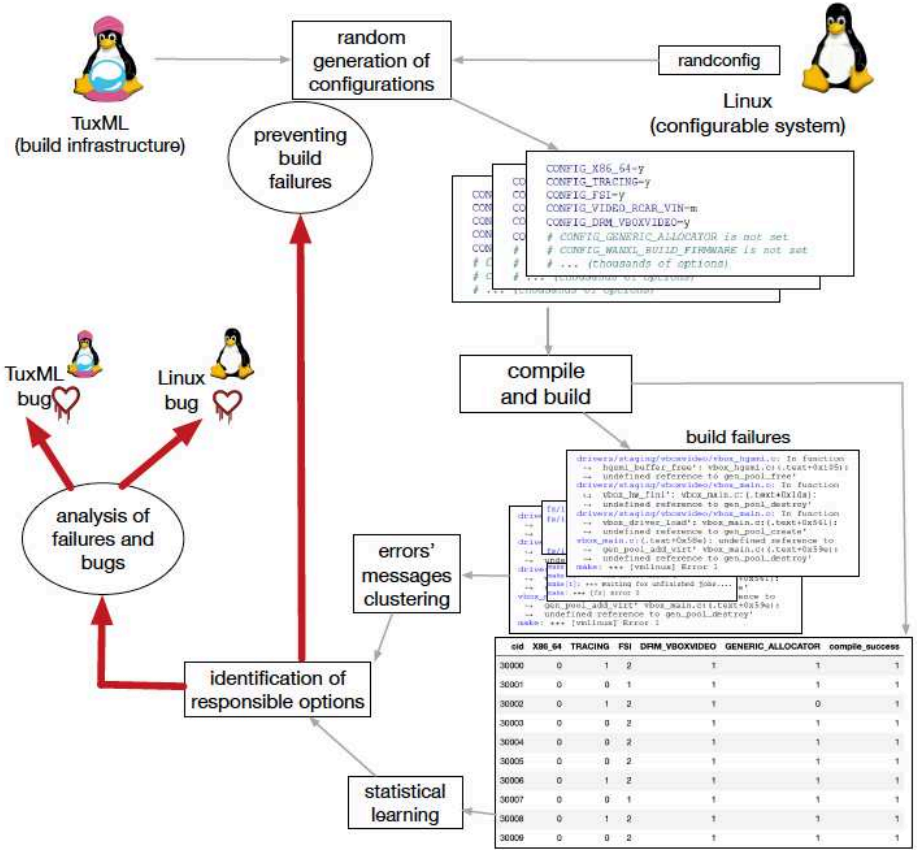
cid ▾ 1	compilation_date	compilation_time	config_file	stdout_log_file	stderr_log_file	user_output_file	compiled_kernel_size	compressed_compiled_kernel_size
74882	2019-08-12 17:09:42	399.856	[BLOB - 24,3 Kio]	[BLOB - 33,7 Kio]	[BLOB - 14 o]	[BLOB - 702 o]	74559280	GZIP-bzImage : 8855504 , GZIP-vmlinux : 10943304 ,...
74881	2019-08-12 16:58:09	460.392	[BLOB - 25,8 Kio]	[BLOB - 34,7 Kio]	[BLOB - 14 o]	[BLOB - 704 o]	81377768	GZIP-bzImage : 18375632 , GZIP-vmlinux : 20462408 ...
74880	2019-08-12 16:47:28	301.775	[BLOB - 22 Kio]	[BLOB - 24,2 Kio]	[BLOB - 14 o]	[BLOB - 705 o]	83004496	GZIP-bzImage : 14365648 , GZIP-vmlinux : 16452424 ...
74879	2019-08-12 16:46:14	1393.61	[BLOB - 24,1 Kio]	[BLOB - 50 Kio]	[BLOB - 571 o]	[BLOB - 712 o]	109098328	GZIP-bzImage : 17183792 , GZIP-vmlinux : 19272160 ...
[illegible]	[illegible]:03	305.705	[BLOB - 26,1 Kio]	[BLOB - 28,8 Kio]	[BLOB - 14 o]	[BLOB - 703 o]	55523752	GZIP-bzImage : 14767568 , GZIP-vmlinux : 16852088

Console de requêtes SQL

74K+ configurations for Linux 4.15

95K+ configurations for Linux 4.13.3

Application 1: “Smart” build infrastructure



Results for 4.13.3

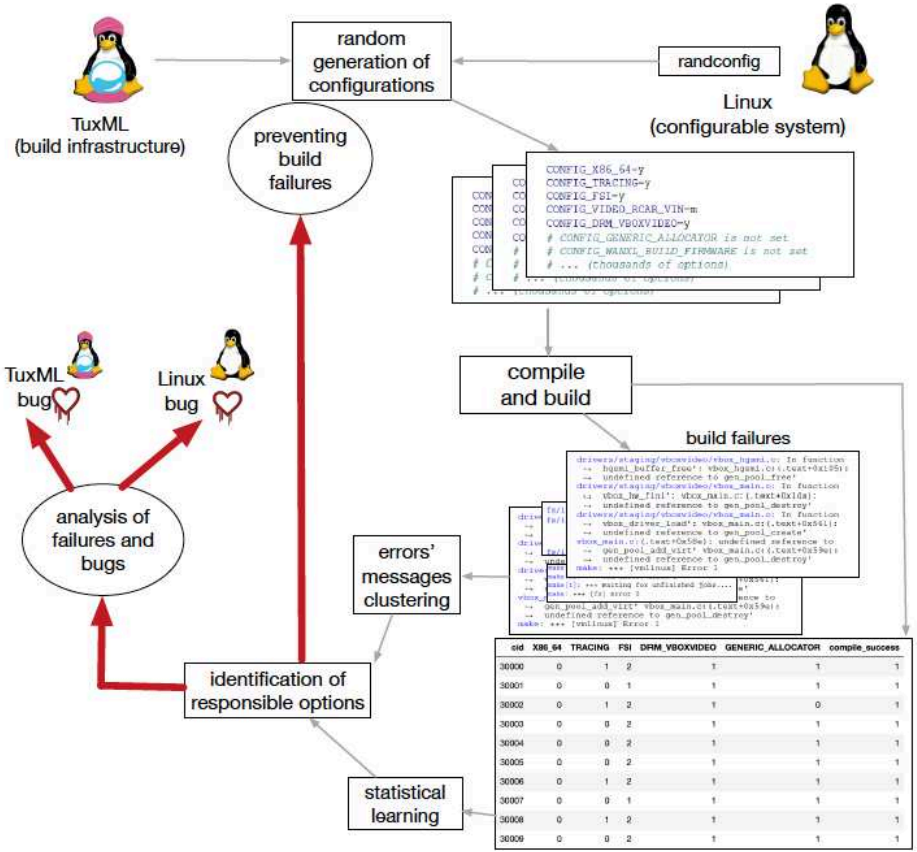
95,854 configurations
3,164 configuration failures
(5.83% of build lead to failures)

Application 1: “Smart” build infrastructure

nb_failures	percentage	bug (faulty option)	Bug?	Fix
2464	68.05	AIC7XXX_BUILD_FIRMWARE ∣ AIC79XX_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
476	13.15	WANXL_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
367	10.14	DRM_VBOXVIDEO & GENERIC_ALLOCATOR	Linux	Kconfig dependency
161	4.45	AIC7XXX_BUILD_FIRMWARE ∣ AIC79XX_BUILD_FIRMWARE	TUXML	missing tools / Kconfig doc.
83	2.29	FORTIFY_SOURCE & UBSAN_SANITIZE_ALL & INFINIBA...	Linux	source code
19	0.52	VIDEO_MUX & VIDEO_V4L2	Linux	Kconfig dependency
15	0.41	BACKLIGHT_CLASS_DEVICE & DRM_I915 ∣ DRM_SAVAGE...	Linux	Kconfig dependency
13	0.36	DRM_VBOXVIDEO & DRM_TTM	Linux	Kconfig dependency
6	0.17	NLS & ..	Linux	source code
3	0.08	SPI_JCORE & ..	Linux	source code
3	0.08	GPIOLIB & ..	Linux	Kconfig dependency
2	0.06	CRC32 & VIDEO	Linux	Kconfig dependency
2	0.06	BT_HCIUART_H4 & ..	Linux	Kconfig dependency
2	0.06	REGMAP_MMIO & ..	Linux	Kconfig dependency
1	0.03	VIDEO_SAA7134_GO7007 & SND_SOC_RT5514_SPI	Linux	Kconfig dependency
1	0.03	USB_F_TCM & ..	Linux	Kconfig dependency
1	0.03	VIDEO_SOLO6X10 & ..	Linux	source code
1	0.03	VIDEO_ATOMISP & ..	Linux	source code + Kconfig dep.
1	0.03	NEW_LEDS & ..	Linux	Kconfig dependency

5.83% of build failures BUT
only due to 16 configuration bugs of Linux and 3 configuration bugs of… TUXML
We come to this insight thanks to our learning procedure

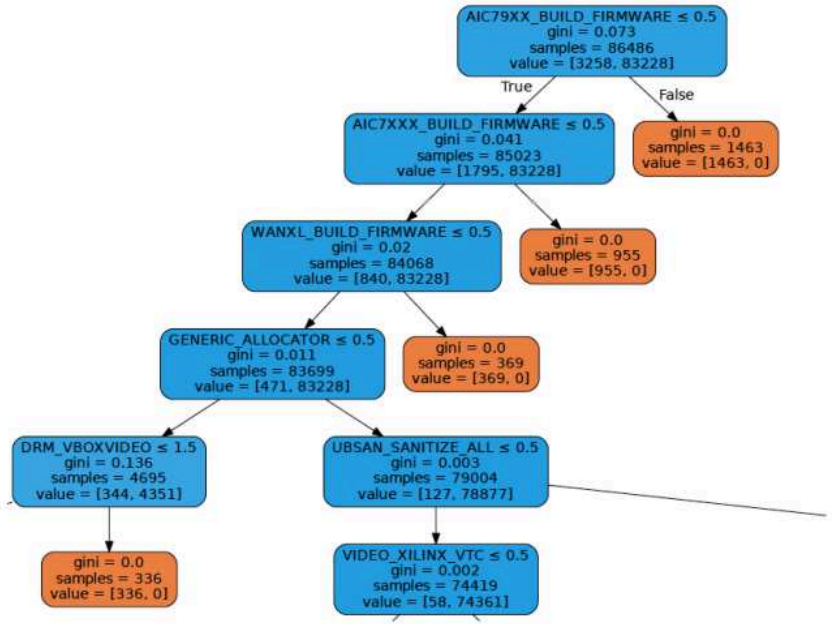
Application 1: “Smart” build infrastructure



5.83% of build failures
BUT
only due to 16 configuration bugs of Linux and 3 configuration bugs of… TUXML

We come to this insight thanks to our learning procedure

Application 1: “Smart” build infrastructure



5.83% of build failures
BUT
only due to 16 configuration bugs of Linux and 3 configuration bugs of… TUXML

We come to this insight thanks to our learning procedure

Application 1: “Smart” build infrastructure

```
X86_64=y
USB=m
...
# always leads to a failure
AIC7XXX_BUILD_FIRMWARE=y
...
# always leads to a failure
DRM_VBOXVIDEO=y
GENERIC_ALLOCATOR=n
```

```
aicasm_symbol.c:48:19: fatal error: aicdb.h: No such
 ↪  file or directory
 #include "aicdb.h"
                   ^
make[4]: *** [aicasm] Error 1
make[3]: *** [drivers/scsi/aic7xxx/aicasm/aicasm]
 ↪  Error 2
make[2]: *** [drivers/scsi/aic7xxx] Error 2
make: *** [drivers] Error 2
```

(a) Configuration failure due to AIC7XXX_BUILD_FIRMWARE

```
X86_64=y
USB=y
...
AIC7XXX_BUILD_FIRMWARE=n
...
# always leads to a failure
DRM_VBOXVIDEO=y
GENERIC_ALLOCATOR=n
```

```
drivers/staging/vboxvideo/vbox_hgsmi.o: In function
 ↪  hgsmi_buffer_free': vbox_hgsmi.c:(.text+0x105):
 ↪  undefined reference to gen_pool_free'
drivers/staging/vboxvideo/vbox_main.o: In function
 ↪  vbox_hw_fini': vbox_main.c:(.text+0x14a):
 ↪  undefined reference to gen_pool_destroy'
drivers/staging/vboxvideo/vbox_main.o: In function
 ↪  vbox_driver_load': vbox_main.c:(.text+0x561):
 ↪  undefined reference to gen_pool_create'
vbox_main.c:(.text+0x58e): undefined reference to
 ↪  gen_pool_add_virt' vbox_main.c:(.text+0x59e):
 ↪  undefined reference to gen_pool_destroy'
make: *** [vmlinux] Error 1
```

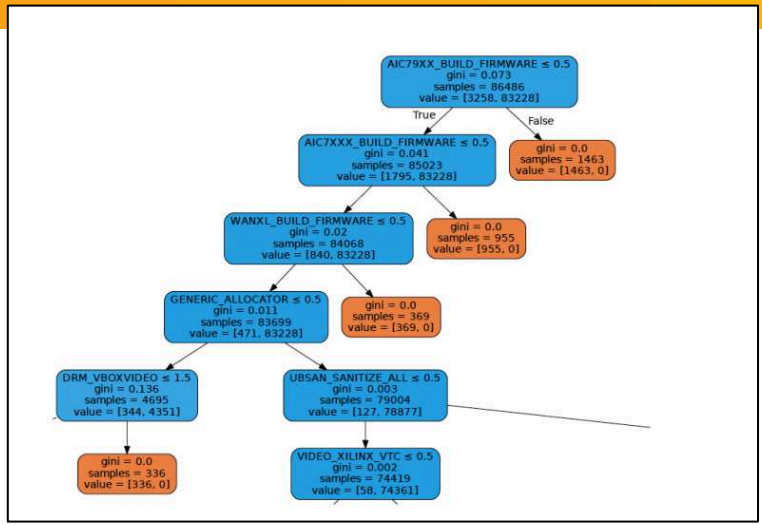
(b) configuration failure due to DRM_VBOXVIDEO, GENERIC_ALLOCATOR

```
X86_64=y
# always leads to a failure
AIC7XXX_BUILD_FIRMWARE=y
# always leads to a failure
FORTIFY_SOURCE=y
UBSAN_NULL=y
UBSAN_SANITIZE_ALL=y
IPV6=y
INFINIBAND_ADDR_TRANS=y
...
```

```
./include/linux/string.h:305:4: error: call to
 ↪  '__read_overflow2' declared with attribute
 ↪  error: detected read beyond size of object
 ↪  passed as 2nd parameter
    __read_overflow2();
    ^~~~~~~~~~~~~~~~~~
make[3]: ***
 ↪  [drivers/infiniband/core/roce_gid_mgmt.o] Error
 ↪  1
make[2]: *** [drivers/infiniband/core] Error 2
make[1]: *** [drivers/infiniband] Error 2
make: *** [drivers] Error 2
```

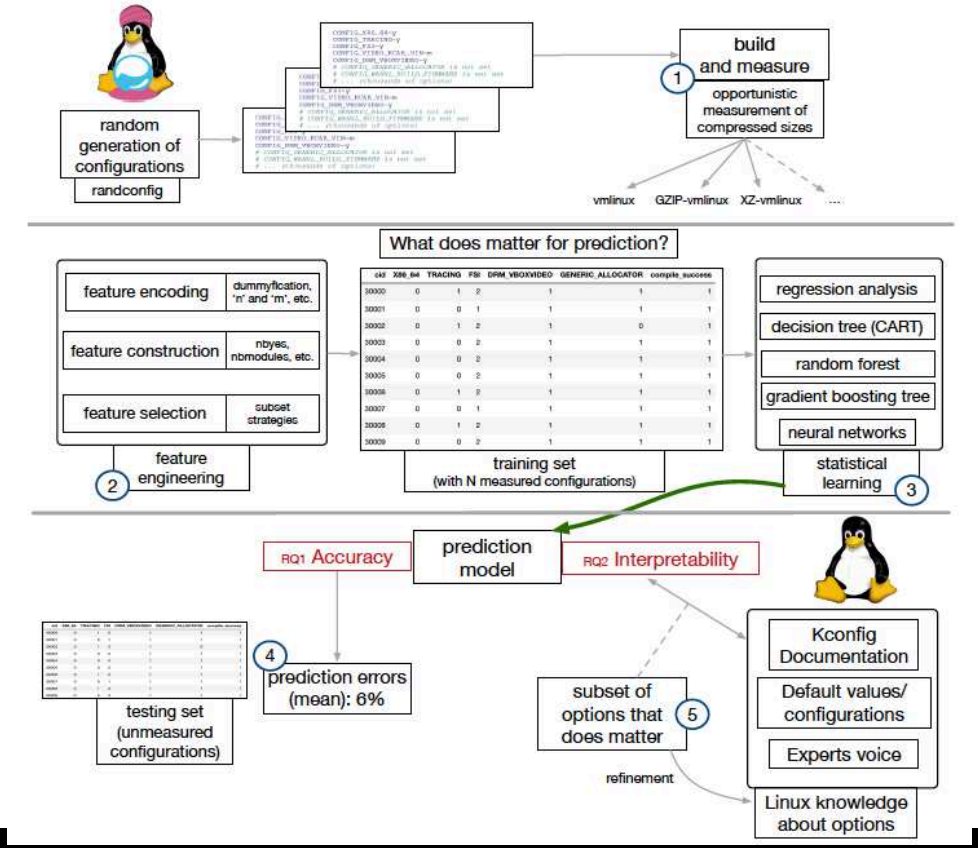
(c) Configuration failure due to FORTIFY_SOURCE +options in red

Figure 2: Configuration failures and error messages. Which options’ values cause the failures? How to prevent failures?

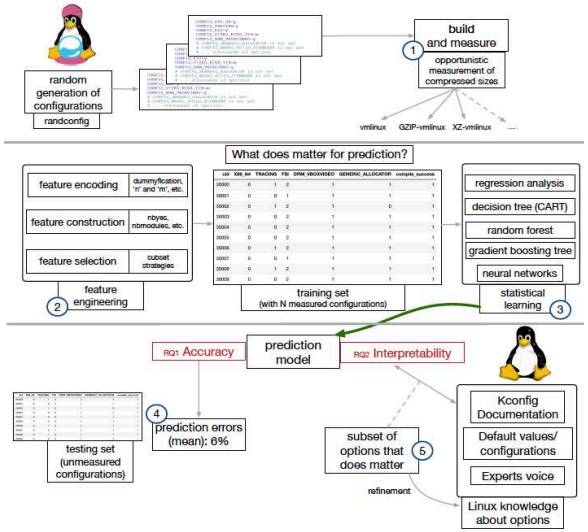


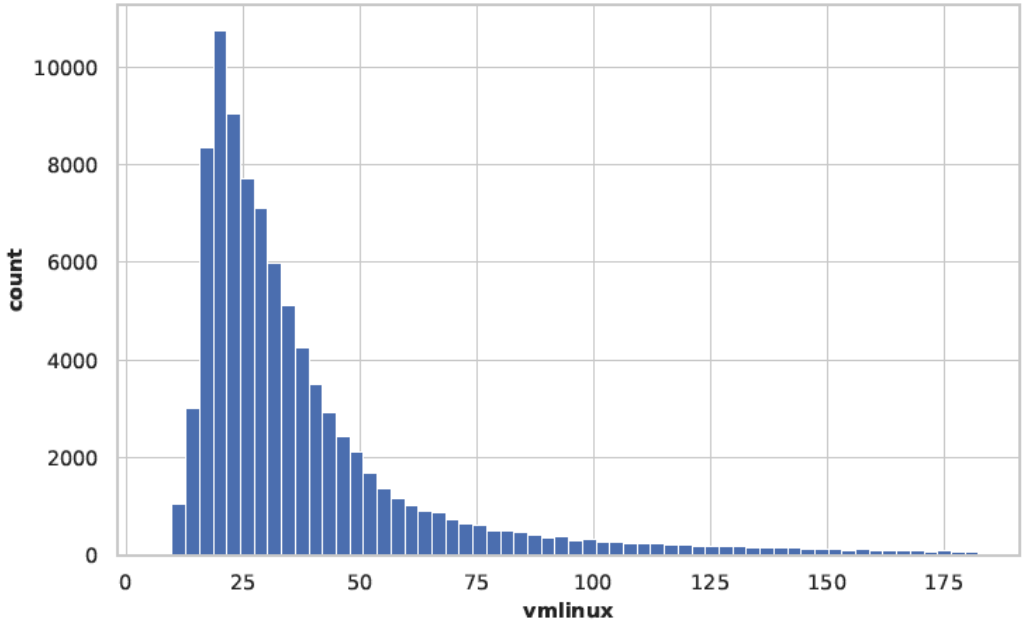
1. **Some configuration failures mask other failures**
2. **Statistical learning is not sufficient (also consider/ cluster failure messages)**
3. **TUXML can prevent failures (including accidental failures)**

Application 2: Kernel Size Prediction



Application 2: Kernel Size Prediction





Max: 1,698.14Mb

Min: 7Mb (tinyconfig)

Figure 4: Distribution of size (in Mb) without outliers

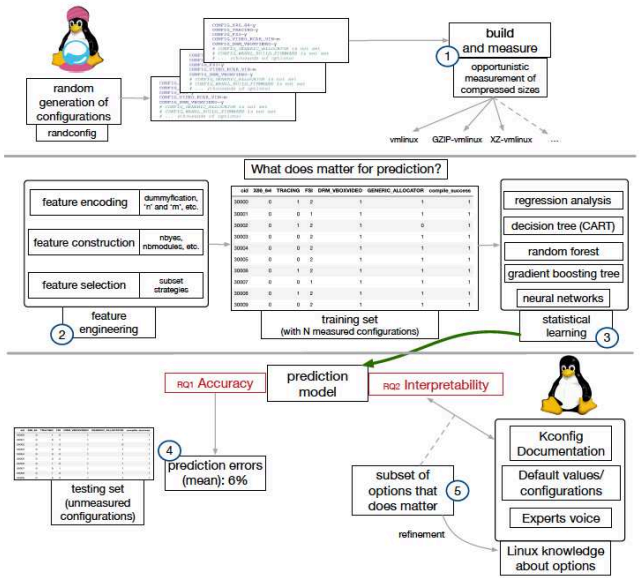
Application 2: Kernel Size Prediction

Algorithm	Without Feature Selection					With Feature Selection				
	N=10	N=20	N=50	N=80	N=90	N=10	N=20	N=50	N=80	N=90
OLS Regression	74.54±2.3	68.76±1.03	61.9±1.14	50.37±0.57	49.42±0.08	43.56±1.48	42.58±2.22	40.23±0.22	39.56±0.39	39.29±0.48
Lasso	34.13±1.38	34.32±0.12	36.58±1.04	38.07±0.08	38.04±0.17	35.18±0.45	36.53±0.6	39.28±1.06	38.28±0.04	38.61±0.81
Ridge	139.63±1.13	91.43±1.07	62.42±0.08	55.75±0.2	51.78±0.14	43.52±1.41	42.29±2.16	40.2±0.27	39.53±0.33	39.24±0.43
ElasticNet	79.26±0.9	80.81±1.05	80.58±0.77	80.57±0.71	80.34±0.53	79.66±2.11	81.74±0.65	81.0±0.24	80.84±0.6	81.45±0.2
Decision Tree	15.18±0.13	13.21±0.12	11.32±0.07	10.61±0.10	10.48±0.15	13.97±0.08	12.34±0.08	10.75±0.05	10.07±0.09	9.91±0.12
Random Forest	12.5±0.19	10.75±0.07	9.27±0.07	8.6±0.07	8.4 ±0.07	10.79±0.15	9.6±0.08	8.4±0.05	7.96±0.06	7.8±0.05
GB Tree	11.13±0.23	9.43±0.07	7.70±0.04	7.02±0.05	6.83±0.10	8.67±0.09	7.60±0.08	6.65±0.03	6.33±0.03	6.24±0.06
N. Networks	16.73 ±1.30	11.38 ±0.27	9.34 ±0.17	8.11 ±0.26	7.76 ±0.10	14.20 ±0.02	8.7 ±0.06	6.61 ±0.02	5.73 ±0.03	5.52 ±0.12
Polynomial Reg.	-	-	-	-	-	24,65±1.23	22.58±0.18	20.49±0.24	21.53±0.1	20.86±0.04

Table 1: MAPE of different learning algorithms for the prediction of vmlinux size, without and with feature selection

We find a sweet spot where only 200—300 features are sufficient to efficiently train a random forest and a Gradient Boosting Tree to obtain a prediction model that outperforms other baselines (7% prediction errors for 40K configurations). We observe similar feature selection benefits for any training set size and tree-based learning algorithms.

Application 2: Kernel Size Prediction



Thanks to our prediction model, we have effectively identified a list of important features that is consistent with the options and strategy of tinyconfig, the Kconfig documentation, and Linux knowledge. We also found options that can be used to refine or augment the documentation.

Challenges

- Kernel CI / 0-day
  - Our focus: testing **configurations** in the large
  - Complementary!
  - **Learning** techniques can be used in both contexts
  - Sharing data
- Unify the force!

Challenges

- “Smart” build infrastructure
  - Other properties (e.g., boot, security)
  - “Transfering” a prediction model for another version (without starting from scratch)
- With devs/contributors in the loop
  - We need knowledge to validate our learning model
  - We need knowledge to apply “smart” sampling
  - We aim to produce actionable knowledge
- TUXML needs you!

Some related work

- Julia Lawall and Gilles Muller “JMake: Dependable Compilation for Kernel Janitors.” In 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017
- Iago Abal, Claus Brabrand, and Andrzej Wasowski “42 variability bugs in the linux kernel: a qualitative analysis”. In ACM/IEEE International Conference on Automated Software Engineering, ASE’14
- Jean Melo, Elvis Flesborg, Claus Brabrand, and Andrzej Wasowski “A Quantitative Analysis of VariabilityWarnings in Linux”. In Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS’16)
- Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki ”Where Do Configuration Constraints Stem From? An Extraction Approach and an Empirical Study” IEEE Trans. Software Eng., 2016
- Minghui Zhou, Qingying Chen, Audris Mockus, and Fengguang Wu “On the Scalability of Linux Kernel Maintainers’ Work”. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)

Thanks!

- DiverSE research team http://diverse-team.fr
  - Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
- TUXML team at ISTIC / University of Rennes 1
  - Paul Saffray, Alexis Le Masle, Michaël Picard, Corentin Chédotal, Gwendal Didot, Dorian Dumanget, Antonin Garret, Erwan Le Flem, Pierre Le Luron, Mickaël Lebreton, Fahim Merzouk, Valentin Petit, Julien Royon Chalendard, Cyril Hamon, Luis Thomas, Alexis Bonnet
- IGRIDA http://igrida.gforge.inria.fr

Preprints (feedbacks welcome!)

- Learning From Thousands of Build Failures of Linux Kernel Configurations
  - Mathieu Acher, Hugo Martin, Juliana Alves Pereira, Arnaud Blouin, Djamel Eddine Khelladi, Jean-Marc Jézéquel
  - https://hal.inria.fr/hal-02147012
- Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes
  - Mathieu Acher, Hugo Martin, Juliana Pereira, Arnaud Blouin, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Luc Lesoil, Olivier Barais
  - https://hal.inria.fr/hal-02314830

Intrigued by Tux logos?

Have a look and don't hesitate to contribute!

https://github.com/diverse-project/tuxart

Side project: Tux generator out of arbitrary Linux kernel configurations (.config)

Khaled Arsalane
Eliot Marie
Pierre Pouteau
Zakariae Boukhchen
Richard Faraji-Huon
Mathieu Acher


Embedded Linux
Conference
Europe